Monte Carlo Simulation, Schuëller & Spanos (eds) © 2001 Balkema, Rotterdam, ISBN 90 5809 188 0

Monte Carlo analysis of structural systems using neural networks

Jorge E. Hurtado & Diego A. Alvarez
Universidad Nacional de Colombia, Manizales, Colombia
Alex H. Barbat
Universidad Politécnica de Cataluña, Barcelona, Spain

ABSTRACT: Monte Carlo simulation is increasingly being used in the analysis of large and complex structural systems for the assessment of the uncertainty spread and the reliability. A major handicap for the popularization of this technology is the large number of deterministic evaluations needed to such purposes, inasmuch as linear or nonlinear finite element solvers are required for each output sample calculation. In order to simplify this task neural networks are evaluated in this paper as a partial surrogate of the deterministic solver. The neural networks are trained with the input/output pairs resulting from a few number of finite element simulations, and are henceforth used in a Monte Carlo context. It is shown that when employed in this way, neural networks constitute a promising tool for a drastic reduction of the computational cost needed by a Monte Carlo simulation in this field of application. Three types of networks have been selected for the study, two of which correspond to supervised and the other one to hybrid learning procedures. The paper compares the network designs in their more relevant aspects, which are the training speed and accuracy, the extrapolation ability and the accuracy of the estimated probabilities.

1 INTRODUCTION

The application of Monte Carlo methods for the reliability analysis of structural systems requires the repeated use of finite element solvers, which is a task implying a high computational cost, especially if a large number of elements are employed in the model. It seems that the fast development of computing speed is not contributing to the popularization of Monte Carlo methods in practical structural design due to the fact that it is commonly accompanied by a parallel increase of the number of finite elements employed to each problem. As a consequence current design practice seems to be confined into the narrow physical frame of deterministic views (Marczyk 1997). Thus the employment of Monte Carlo techniques in practical structural design seems to be continuously postponed.

This paper explores a way for partial alleviation of the mentioned computational effort that can be useful even in complex cases modelled with thousands of finite elements. This consists in using neural networks as partial surrogate of the numerical mechanical model. Such an approach is suggested by the notorious association and memory properties of these devices.

As is known, Artificial Neural Networks (ANN) constitute one of the most important recent developments in the field of Artificial Intelligence together with Genetic Algorithms and Fuzzy Logic. In essence, the ANN are intended to imitate the learning and reasoning mechanisms of living beings. They have proved to be very useful for multiple engineering purposes, such as pattern recognition, image and signal processing, optimization, automatic control, etc (Lin & Lee 1996). Also, they have found numerous applications in computational and experimental mechanics, as summarised by Yagawa & Okuda (1996).

A real biological neuron is composed by a cell and several dendrites, which connect it to other neurons. The activity of the neuron comprises the following phases: a) the reception of the electrical impulses sent to it by other neurons; b) the transformation of the collected information and c) the emission of the processed impulse to other neurons. This process has been imitated both in software and hardware forms for different technical purposes. In applying the ANN to many problems of structural mechanics, however, only the algorithms for training and using the ANN are of importance. In this case they are explored in the structural reliability context. As is well known, this problem can be expressed as

$$P_{\mathrm{f}} = \int_F f(\boldsymbol{X})\mathrm{d}\boldsymbol{X} \tag{1}$$

where P_{f} is the probability of exceeding a critical response, $\boldsymbol{X}$ is the vector of random structural parameters and F denotes the mapped failure domain in the $\boldsymbol{X}$-space. This can be formulated as

$$F = \{\boldsymbol{X} : g(\boldsymbol{X}) \leq 0\} \tag{2}$$

where $g(\boldsymbol{X})$ is the limit state function. There are roughly two families of methods to solve the reliability problem, namely, the analytical approximation of the above integral by First or Second Order Reliability Methods (usually denoted as FORM and SORM) and integration techniques (Melchers 1999). Although analytical techniques are computationally simple they are nonetheless limited to cases with a smooth limit state function and are otherwise inaccurate (Schuëller and Stix 1987). On the other hand, the numerical estimation of the integral can be performed by Monte Carlo methods, which are general with respect to the shape of the limit state function. In lieu of approximating the failure domain by simple functions in the $\boldsymbol{X}$ space, Monte Carlo methods allow the direct computation of the failure probability by generating an artificial population of structural responses $g(\boldsymbol{X})$ using a deterministic finite element code (Melchers 1999). This requirement, of course, implies a large computational cost which can become prohibitively high when the probability of failure is low. It is just for this task that neural networks are useful, given their ability to learn the mapping of input onto output variables of a given system regardless of its physical or probabilistic models. The computation of failure probabilities via ANN comprehends the following phases:

1. To calculate several input/output pairs to learn the network via standard Monte Carlo techniques using the finite element solver.
2. To train the network with these pairs.
3. To use the trained network as a solver surrogate for the reliability assessment in the subsequent Monte Carlo simulations.

In this respect it must be observed that ANN approach resembles the Response Surface method (Faravelli 1989, Kim & Na 1997) in that both are purported to substituting the deterministic code used for Monte Carlo integration. However, the Response Surface method relies on the validity of a simple equation, usually of polynomial form, which is fitted to the preliminary data by least squares techniques. On the contrary, neural networks provide a more general approach in that the mapping functions require a much larger number of parameters, allowing the introduction of a high complexity degree that can cope with the inherent nonlinearities of the problem at hand. In addition, notice that with the goal of memory association the training samples need not be drawn exclusively from the given density functions of the basic variables. Instead they can be arbitrarily generated. This implies a significant advantage over advanced Importance Sampling-based techniques (Melchers 1999), in whose employment difficulties are usually found for obtaining realizations in the failure region required for assigning an adequate Importance Sampling density (Ang et al. 1991).

The following section contains a brief summary of the basic features of the training of the two types of networks used to this study. Then follows a discussion on the practical issues of the training and production phases for reliability computations using a structural example as a guide, characterised by a nonlinear limit state function.

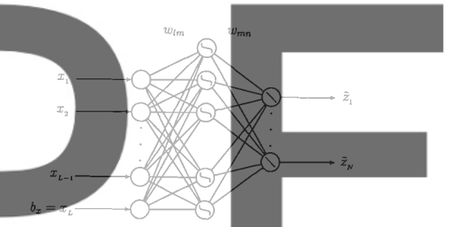

Figure 1. Backpropagation multilayer perceptron.

2 NEURAL NETWORK TYPES

In this section the main features of the two network types, namely, the Backpropagation (BP) Multi-Layer Perceptrons and the Radial Basis Function Networks (RB) are described, as coded in the software MATLAB (The Math Works 1994), which was used in the investigation.

2.1 *Backpropagation Multi-Layer Perceptrons*

This kind of neural networks imitate certain memorizing processes performed by living beings in the following form:

1. The information arriving to a neuron j from the neuron l, x_{lj}, is multiplied by a weight w_{lj}, which represents the importance of the latter on the activity of the former among all the neurons transferring information to it.
2. The total information given to neuron $m, m = 1, 2, \ldots, M$ is calculated by the following weighted sum:

$$\bar{y}_m = \sum_{l=1}^{L} w_{lm} x_l \tag{3}$$

3. The result is transformed by a linear or a nonlinear function:

$$\tilde{y}_j = f(\bar{y}_j) \tag{4}$$

4. The transformed information is sent to neuron k using the corresponding weights w_{jk}.

Linking several layers of neurons in this way, one one can create an entire artificial multi-layer perceptron as illustrated by Fig. 1 for the case of three layers. Notice the presence of a bias term $b_x = x_L$ which is usually set to 1. The Backpropagation procedure of training the network comprises these steps: (a) the information on input data is entered through the first layer, usually in a normalized form; the connection weights, which are the objective of the training, are usually set equal to random numbers at the first stage; (b) at the second (or hidden) layer the weighted sum (3) and the nonlinear transformation (4) are performed; (c) the same procedure is operated at the third (or output) layer, which has as many neurons as output data. Since the result given by these neurons differ from the known results corresponding to the input data, there is a need to updating the weights of the whole network upon the basis of the error in the output sets in the backward direction. When the weights are obtained by the condition that the square error be a minimum, the procedure is known in the ANN literature as the *delta rule* (DR). This algorithm is considered to be of the *supervised* type because the error with respect to known outputs are used in calculating all the connection weights.

Several alternative proposals to the above minimization procedure have been published. One of them employs the Levenberg-Marquardt method (LM), in which the weight vector is updated according to the following equation:

$$\boldsymbol{w}[k+1] = \boldsymbol{w}[k] - (\boldsymbol{J}\boldsymbol{J}^{\mathrm{T}} + \lambda \boldsymbol{I})^{-1} \boldsymbol{g} \tag{5}$$

where $\boldsymbol{J}$ is the Jacobian matrix of the ANN outputs with respect to the weigths, $\boldsymbol{I}$ is the identity matrix and λ is a suitable nonnegative value.

Finally, it must be said that usually is necessary to repeat the training several times to reach a low level of error when using any of the above two algorithms. In each of these training *epochs* the data set can be presented to the network in the same order or in a randomized fashion.

In this paper the classical delta rule and the Levenberg-Marquardt optimization technique have been tested.

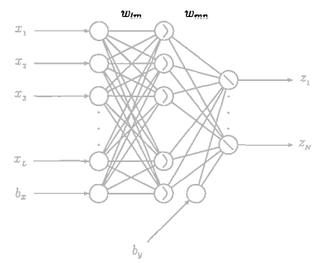

Figure 2. Radial Basis Function Network.

2.2 *Radial Basis Function Networks*

The Radial Basis Function networks (RB) have their inspiration in the associative properties of the biological neurons. It differs from the BP type in that it is intended to mimic an input–output mapping not by building a complex nonlinear relationship but by classifying the data into several groups. Besides being a common task in the field of Pattern Recognition (Ripley 1996, Theodoridis & Koutroumbas 1999), this feature can also be exploited in the reliability analysis context, because the limit state functions define two well distinguished regions, namely the safe and failure ones. In numerical practice, the most relevant advantage of this network type is that there is no need of presenting the training set hundreds or thousands times, as is commonly required by the Backpropagation strategies summarized above. Instead, the network can be readily calculated as a simple least squares problem. This opens up the possibility of a fast computation of several alternatives in order to select the most accurate for the given training set.

The architecture of the RB network adopted in MATLAB (The Math Works 1994) is described in Fig. 2. The first layer represents the given input values $x_l, l = 1, 2, \ldots, L$, to which a bias neuron b_x is added. The weights in the first layer are defined as a different input vector selected at random from the database. In addition, the bias term in the first layer b_x equals

$$b_x = \sqrt{-\ln 0.5}/\Omega \tag{6}$$

where Ω is a constant determining the amplitude of the Radial Basis function. This constant serves to identify each network model, as it is not a target of the calculation. Henceforth this proceeds according to the following steps:

1. The input to the second layer $\bar{y}_m$ is the Euclidean distance between inputs and weights:

$$\bar{y}_m = \sum_{l=1}^{L} ||w_{lm} - x_l|| \qquad (7)$$

Since the weights are in fact equal to other inputs, this distance measures how close they are to each other.

2. The Radial Basis function used in this layer has a bell shape

$$\tilde{y}_m = f(\bar{y}_m) = \exp\left(-\frac{\bar{y}_m^2}{2}\right) \qquad (8)$$

Accordingly, this function privileges those inputs which are the nearest to the given one.

3. The outputs of the second layer as well as a new bias term b_y are multiplied by weight factors w_{mn} . The input to the third layer is

$$\bar{z}_n = \sum_{m=1}^{M} w_{mn}\tilde{y}_m \qquad (9)$$

4. A purely linear transformation is applied to the input of the third layer so that $\tilde{z}_n = \bar{z}_n$.

5. The weights and the bias in the second layer are calculated by a least squares problem set by the outputs of the second layer and the known outputs of the third.

This algorithm is considered as *hybrid* between the supervised an unsupervised types because only some of the weights are computed on the basis of the errors with respect to known outputs.

3 NUMERICAL DEMONSTRATION

In order to examine the capabilities of the three network algorithms summarized above, a numerical study was conducted over a frame structure depicted in Fig. 3. The response under analysis is the top horizontal displacement of the frame. The critical value was chosen as 11 cm.

In both the BP and RB types a purely linear function was employed in the output layer. In the two BP algorithms use was made of an adaptive learning rate η and momentum updating. In the RB case, dynamic adaptation is implemented by adding new neurons to the hidden layer as new data are presented to the network. The three types of networks were tested using nine models, according to the number of neurons M in the hidden layer for the BP types and to the spread constant Ω in the RB case. The models are defined in Table 2. For each model 20 analysis were performed and the best one in the sense of least end training error was selected as a representative of its category.

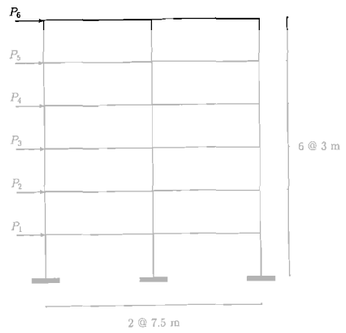

Figure 3. Frame example.

Table 1. Example 2 - Probability distributions of the basic variables

X	$f_X(x)$	μ	σ
E, cols.	LN	1960 kN/cm^2	196 kN/cm^2
E, beams	LN	1960 kN/cm^2	196 kN/cm^2
I, cols.	LN	100,000 cm^4	10,000 cm^4
I, beams	LN	150,000 cm^4	15,000 cm^4
P_1	N	24.50 kN	6.125 kN
P_2	N	27.44 kN	6.860 kN
P_3	N	28.42 kN	7.105 kN
P_4	N	29.40 kN	7.350 kN
P_5	N	30.38 kN	7.595 kN
P_6	N	31.36 kN	7.840 kN

The networks were trained with sets of 150, 600 and 1000 samples. After training, the failure probabilities were calculated with 25,000 samples. The random variates were combined by means of the Updated Latin Hypercube method (Florian 1992). The total number of trainings performed for this example was therefore equal to 420. This large number allows a good comparison of the network design and training options with respect to the following aspects:

1. Training time and mean error.

2. Consistency of the estimation of the probability of failure.

A standard Monte Carlo analysis was performed over the actual structural model for calculating the horizontal displacements. This implies the solution of the system

Table 2. Neural network models tested

Model	DR M	LM M	RB $10^{-5}\Omega$
1	2	2	1.00
2	3	3	1.96
3	4	4	2.92
4	5	5	3.88
5	6	6	4.84
6	7	7	6.44
7	8	8	7.40

$$\boldsymbol{K}\boldsymbol{u} = \boldsymbol{p} \qquad (10)$$

where $\boldsymbol{K}$ is the stiffness matrix of the structure, which is condensed with respect to the desired degrees of freedom; $\boldsymbol{u}$ is the displacement vector and $\boldsymbol{p}$ the load vector. The probability of failure was estimated with 350 000 calls of the finite element solver as 0.005451. This figure will be used as the exact value in the following comparisons.

3.1 *Training time and mean error*

Figure 4 compares the training times for the nine models of the BP networks when using 150 samples, as measured on a standard PC. It can be seen that the training time of the BP networks depends almost linearly on the number of neurons. If this were the only variable to consider, these results suggest that for complex problems involving many basic variables the LM training algorithm would be preferred over the classical delta rule. Finally, it can be seen that the training time of the RB architecture is not sensitive to the spread constant as can be expected.

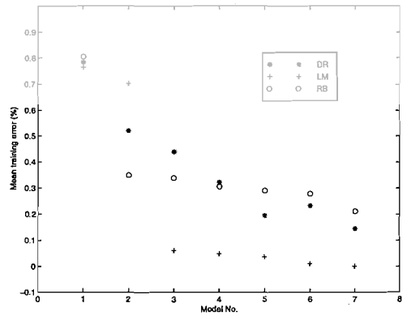

Figure 4. Training times with 150 samples.

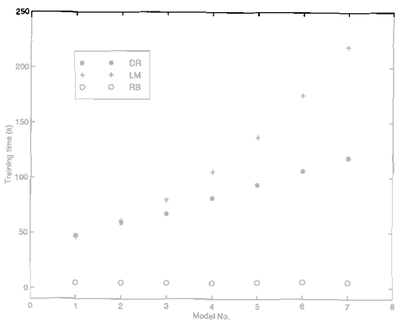

Figure 5. Mean training errors with 150 samples.

On the other hand, Figure 5 shows the end mean error in percent for the two BP cases, as measured with respect to the training set. It can be observed that in both cases the increase of the number of neurons in the hidden layer has a positive effect on the error. Such an increase is more effective for the LM that for the DR algorithm. In this respect, it must be noticed that the generalization capacity of the network can be negatively affected by an uncontrolled increase of this number, as is well established (Lin & Lee 1996).

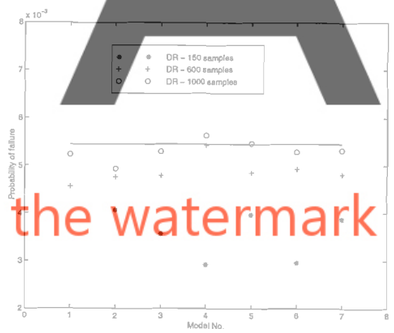

Figure 6. Performance of the DR network.

3.2 *Consistency of the estimation of the probability of failure*

Important as it is, however, the comparison of end training error is not conclusive about the optimal network type because two important questions remain, namely,

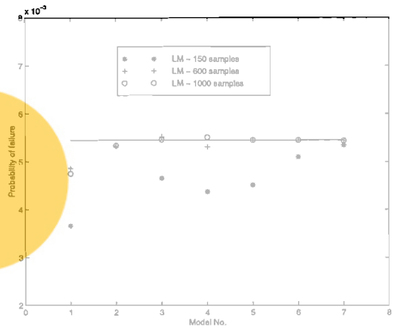

Figure 7. Performance of the LM network.

1. The extrapolation capacity of the network, i.e. its accuracy in assessing the limit state values and hence the failure probability.

2. The consistency of the network type when tested with the validation set. In other words, the robustness of the estimations of the probability of failure given by the network type with respect to changes of the parameters defining the model and of the sample size.

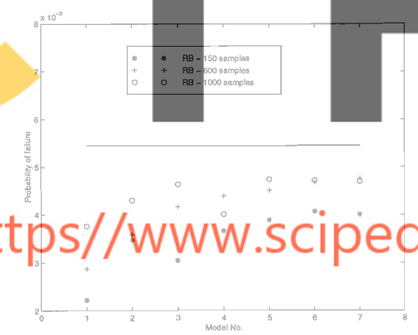

Figure 8. Performance of the RB network.

Figures 6 to 8 show the performance of the three network types in terms of the estimated failure probabilities. The value calculated by direct Monte Carlo is shown as a straight line. It can be seen that the DR network does not perform as well as the LM one, which often gives the correct value. Besides, the dispersion of the LM results are much lower than those of the DR type and its convergence to the right value is faster. In both cases the extrapolation capacity of the networks (i.e. their ability to give good estimations of outputs not used in the learning process) is not affected by increasing the number of hidden neurons. In all cases the RB network underestimates the failure probability even with a large training set. Moreover, the improvement of the estimation seems to saturate when increasing the sample size. This can perhaps be attributed to the nonlinearity and the high dimensionality of the limit state function. This contrasts with the excellent results given by this network type when tested on linear limit functions (not reported herein). Thus it seems less robust than the LM algorithm for general limit states.

4 CONCLUSIONS

Two classes of neural networks have been examined as surrogates of the structural model needed in a Monte Carlo assessment of its probability of failure. They are the Backpropagation Multi-Layer Perceptron and the Radial Basis Function network, which correspond respectively to supervised and hybrid learning types. The first was tested using two training approaches, namely the popular Delta Rule and the Levenberg-Marquardt minimization procedure. The following conclusions can be drawn from this study:

1. The LM algorithm behaves better than the DR in terms of training time and error, accuracy and robustness.

2. The effectivity of the RB network seems to saturate when increasing the training sample size.

3. The BP-LM algorithm is the one that can be more safely recommended for practical use. To that purpose it is suggested that it be trained several times with different numbers of neurons in the hidden layer and several training samples sizes in order to select the statistical mode of the results as the final estimation. The effect of this procedure over the computational labor is negligible as the training times of the networks are very low and no new finite element solver calls are needed for each training.

REFERENCES

Ang, G.L., A.H.S. Ang and W.H. Tang, 1991, Optimal Importance Sampling Density Estimator. *Journal of Engineering Mechanics*, 118: 1146-1163.

Chapman, O.J. and A.D. Crossland, 1995, Neural Networks in Probabilistic Structural Mechanics, in *Probabilistic Structural Mechanics Handbook*, edited by C. (Raj) Sundararajan. Chapman and Hall, New York.

Faravelli L., 1989, Response-surface approach for reliability analysis. *Journal of Engineering Mechanics*, 115: 2763-2781.

Florian, A., 1992, An Efficient Sampling Scheme: Updated Latin Hypercube Sampling, *Probabilistic Engineering Mechanics*, 7: 123-130.

Hong, H.P. and N.C. Lind, 1996, Approximate reliability analysis using normal polynomial and simulation results, *Structural Safety*, 18: 329-339.

Hurtado, J.E. and A.H. Barbat, 1998, Monte Carlo Techniques in Computational Stochastic Mechanics, *Archives of Computational Methods in Engineering*, 5: 3-30.

Kim, S.H. and Na, S.W., 1997, Response surface method using vector projected points. *Structural Safety*, 19:3-19

Lin, C.T. and G. Lee, 1996, *Neural Fuzzy Systems* Prentice-Hall, Upper Saddle River.

Long, F.L. and R. Unbehauen, 1998, *Applied Neural Networks for Signal Processing*. Cambridge University Press, Cambridge.

Marczyk, J. 1997, Meta-computing and computational stochastic mechanics, in *Computational Stochastic Mechanics in a Meta-computing Perspective*, edited by J. Marczyk. CIMNE, Barcelona.

Melchers, R.E., 1999, *Structural Reliability: Analysis and Prediction*, 2nd Edition. John Wiley and Sons, Chichester.

Ripley, B.D., 1996, *Pattern Recognition and Neural Networks*. Cambridge University Press, Cambridge.

Rubinstein, R.Y., 1981, *Simulation and the Monte Carlo Method*, John Wiley and Sons, New York, 1981.

Schuëller, G.I, and Stix R., 1987, A critical appraisal of methods to determine failure probabilities. *Structural Safety*, 4: 293-309.

The Math Works, Inc., 1994, *MATLAB*. The MathWorks, Natick, MA.

Theodoridis, S. and K. Koutroumbas, 1999, *Pattern Recognition*. Academic Press, London.

Yagawa, G. and H. Okuda, 1996, Neural networks in Computational Mechanics, *Archives of Computational Methods in Engineering*, 3: 435-512.

PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON MONTE CARLO SIMULATION
MONTE CARLO/PRINCIPALITY OF MONACO/18 – 21 JUNE 2000

Monte Carlo Simulation

Contributions in honor of the seventieth birthday
of Masanobu Shinozuka on December 23, 2000

Professor M.Shinozuka

Edited by
G.I.Schuëller
Leopold-Franzens University, Innsbruck, Austria
P.D.Spanos
Rice University, Houston, Texas, USA

A.A.BALKEMA PUBLISHERS LISSE / ABINGDON / EXTON (PA) / TOKYO

Monte Carlo Simulation, Schuëller & Spanos (eds) © 2001 Balkema, Rotterdam, ISBN 90 5809 188 0

Cover: The procedure of Monte Carlo Simulation is based on the game of chance. For this reason it was named by its developers after the world famous Casino of Monte Carlo, located in the principality of Monaco.
Photo: Courtesy of Monaco Information Center, Düsseldorf, Germany.

The texts of the various papers in this volume were set individually by typists under the supervision of either each of the authors concerned or the editor.

Published by: A.A.Balkema, a member of Swets & Zeitlinger Publishers
www.balkema.nl and www.szp.swets.nl

ISBN 90 5809 188 0

Printed in the Netherlands

Table of contents